sensible

APLUS

STRUCTURED APPLESOFT-BASIC PRE-COMPILER

BY
CHARLES HARTLEY

software

T.M.

SENSIBLE SOFTWARE, INC.
6619 PERHAM DRIVE
WEST BLOOMFIELD
MICHIGAN 48033
(313) 399-8877

10.30.8

STRUCTURED APPLESOFT-BASIC PRE-COMPILER

APLUS is a 4K machine-language utility that adds the following structured-programming commands to Applesoft basic: WHEN...ELSE...FIN, UNTIL, WHILE, UNLESS, CASE, SELECT(variable), and (OTHERWISE). Multi-line IF...FIN statements are also supported. APLUS also allows the use of "named" subroutines or "procedures". A programmer can now instruct a program to "DO CURVE-FIT", without worrying about the location of the "TO CURVE-FIT" subroutine. APLUS automatically indents "&LIST"ed programs to clarify the logic flow. The APLUS "&CONVERT" command replaces the above structured-programming commands with "GOTO"'s and "GOSUB"'s to provide a STANDARD Applesoft program as output.

New programs can now be written using "GOTO"-less logic. Existing Applesoft programs can be maintained using APLUS without special conversion.

Hardware:	APPLE II or APPLE II PLUS Language card is optional
Memory:	16K required
Language:	ROM or RAM APPLESOFT
DOS:	Compatible with any version
Includes:	Diskette or Tape & User Manual
Price:	$25.00
From:	

sensible
6619 Perham Drive
West Bloomfield
Michigan 48033
software
T.M

(313) 399-8877

APPLE is a registered trademark of APPLE Computer Company

APPLESOFT-PLUS (APLUS) Structured Basic Pre-compiler

APLUS is a 4K-byte, assembly-language program designed to allow structured-programming commands to be used within APPLESOFT IIa Basic programs. APLUS provides automatically indented listings, logic verification with error messages, and the ability to create an executable, "standard" APPLESOFT program from the APLUS structured-basic program.

The following structured-programming commands are supported:

```
IF...FIN
UNLESS...FIN
WHEN...ELSE...FIN
WHILE...FIN (w/ test before loop)
UNTIL...FIN (w/ test after loop)
FOR...NEXT
CASE...(case1)...(case2)...(OTHERWISE)...FIN
SELECT(variable)...(value1)...(value2)...(OTHERWISE)...FIN
"DO named-procedure"
"TO define-named-procedure"...FIN
```

<u>To Load the APLUS Utility</u>

The APLUS utility is first loaded into memory locations $3Ø3Ø to $3FFF and then relocated up to the highest available free RAM memory space available in your system. This is accomplished with the following commands:

```
Disk systems:  ]BRUN APLUS
Tape systems:  ]CALL -151
               * 3010.3FFFR  3010G
```

If your system has RAM APPLESOFT and 2ØK RAM memory, then you <u>must</u> issue the following "HIMEM" command before loading APLUS:

```
               ]HIMEM: 16*1Ø24
```

After APLUS has been loaded, you can use the following two APLUS commands to list programs and to convert APLUS structured-basic commands into APPLESOFT "GOTO"'s:

```
               ]&LIST
               ]&CONVERT
```

APLUS only checks the first letter, so that "&L" can be used instead of "&LIST" and "&C" can be used instead of "&CONVERT".

The above commands will be available as long as the "FP" and the "HIMEM" commands are not used. The HIMEM-pointer was automatically lowered to protect "APLUS" when the APLUS program was loaded. Use of the "FP" or the "HIMEM" command will move the HIMEM pointer and may allow the APLUS utility to be written over. The "MAXFILES" command may also "write over" APLUS if used.

Creating APLUS Source Programs

The APLUS structured-basic commands can be entered into any APPLESOFT program at any time, regardless of whether APLUS is presently loaded into memory. APPLESOFT programs with APLUS commands imbedded in them can be treated like any other APPLESOFT programs (saved to tape or disk, listed, printed, edited with the standard curser controls, etc.) except that APLUS programs cannot be "RUN" until they have been "&CONVERTED" to standard APPLESOFT.

The normal procedure for using APLUS would be to load APLUS and then load your program or enter it from the keyboard. The "&LIST" command will help to identify logic errors by automatically indenting the listing and by flagging erroneous use of APLUS commands. Once the program is completed (or ready to test), the APLUS source program should be saved to tape or disk. The "&CONVERT" command will then convert the APLUS source program in memory to a "standard" APPLESOFT program in memory. The standard APPLESOFT program can then be saved to tape or disk and/or executed. Line numbers and variable names are the same in both the APLUS source program and the resultant "standard" APPLESOFT program to assist with program debugging.

The "standard" APPLESOFT program produced by APLUS can be optimized using the APPLESOFT-Basic Optimization Library available from SENSIBLE SOFTWARE to remove the remarks and extra colons created during the conversion process to produce a final, production copy of your program.

APLUS Structured-basic Commands

In the following discussion, "L.E." will refer to a LOGICAL EXPRESSION. A LOGICAL EXPRESSION is any expression that would normally be allowed in the "standard" APPLESOFT "IF.....THEN....." statement <u>between</u> the "IF" and the "THEN". For example:

```
    Valid LOGICAL EXPRESSIONS:
    A<B
    (A<B) AND (A=5)
    A   (where A is either Ø or 1)
    Invalid LOGICAL EXPRESSIONS:
    A=B+C
    GOTO 2Ø
    PRINT "HI"
```

LOGICAL EXPRESSIONS must be enclosed in parenthesis to guarantee proper operation in APLUS commands. Failure to enclose L.E.'s in parenthesis will usually result in an "FORMULA TOO COMPLEX" error when you "RUN" your "&CONVERTED" program.

The term COMMANDS will refer to one <u>or</u> <u>more</u> lines of "standard" APPLESOFT code. These lines of code may also contain APLUS commands with one restriction. The restriction is that all "lower" levels of logic must be "terminated" within the lines of code designated by the term COMMANDS. An appropriate error message will be generated if the logic is not properly terminated with "FIN"'s.

The "IF" Command:

Standard APPLESOFT "IF" statements can be used freely in APLUS programs and will not be affected by the "&CONVERT" command.

APLUS adds a multi-line IF statement where one or more lines of COMMANDS following the IF statement will be executed if the LOGICAL EXPRESSION is true:

```
Format            Example           &CONVERTed
IF( L.E. )        IF( A<B )         IF NOT ( A<B ) GOTO N
: COMMANDS        : PRINT           PRINT
::FIN             ::FIN             N REM
```

Notes: The colons preceding each line in the above example should not be entered into your program. They will automatically be generated when you "&LIST" your program.

The "UNLESS" Command:

The UNLESS command corresponds to a "IF NOT" statement. The COMMANDS following the UNLESS statement are only executed if the LOGICAL EXPRESSION is not true.

```
Format            Example           &CONVERTed
UNLESS( L.E. )    UNLESS( A<B )     IF ( A<B ) GOTO N
: COMMANDS        : PRINT           PRINT
::FIN             ::FIN             N REM
```

Notes: The colons preceding each line in the above example should not be entered into your program. They will automatically be generated when you "&LIST" your program.

The "WHEN...ELSE...FIN" Command:

The WHEN/ELSE commands allow you to perform one set of COMMANDS when the LOGICAL EXPRESSION is true and a separate set of COMMANDS when the LOGICAL EXPRESSION is false. The COMMANDS on the lines following the WHEN statement and preceding the FIN statement (that corresponds to the WHEN) will be executed when the LOGICAL EXPRESSION is true. The COMMANDS on the lines following the ELSE statement and preceding the FIN statement (that corresponds to the ELSE) will be executed when the LOGICAL EXPRESSION is false. Each WHEN command _must_ have a matching ELSE...FIN pair following the FIN statement corresponding to the WHEN.

```
Format             Example             &CONVERTed
WHEN( L.E. )       WHEN( A<B )         IF NOT ( A<B ) GOTO N
: COMMANDS         : PRINT"TRUE"       PRINT"TRUE"
::FIN              ::FIN               GOTO M
ELSE               ELSE                N REM
: COMMANDS         : PRINT"FALSE"      PRINT"FALSE"
::FIN              ::FIN               M REM
```

Notes: The colons preceding each line in the above example _should not be entered into your program_. They will automatically be generated when you "&LIST" your program.

The "WHILE" command:

The COMMANDS on the lines between the "WHILE" statement and the corresponding "FIN" will be executed as long as the LOGICAL EXPRESSION is true. This is a test-BEFORE-performing-the-loop type of loop. If the LOGICAL EXPRESSION is false before reaching the WHILE statement, the COMMANDS in the loop will not be executed.

Format	Example	&CONVERTed
WHILE(L.E.)	WHILE(A<B)	IF NOT (A<B) GOTO N
: COMMANDS	: PRINT"LOOP"	PRINT"LOOP"
::FIN	::FIN	N REM

Notes: The colons preceding each line in the above example should not be entered into your program. They will automatically be generated when you "&LIST" your program.

The "UNTIL" Command:

The COMMANDS on the lines between the "UNTIL" statement and the corresponding "FIN" will be executed as long as the LOGICAL EXPRESSION is false. This is a test-AFTER-performing-the-loop type of loop. The COMMANDS in the loop will always be executed at least once!

Format	Example	&CONVERTed
UNTIL(L.E.)	UNTIL(A<B)	N REM
: COMMANDS	: PRINT"LOOP"	PRINT"LOOP"
::FIN	::FIN	IF NOT (A<B) GOTO N

Notes: The colons preceding each line in the above example should not be entered into your program. They will automatically be generated when you "&LIST" your program.

The "FOR...NEXT" Command:

The standard "FOR" and "NEXT" commands are supported by APLUS. APLUS will attempt to indent to show the loop generated by a FOR...NEXT combination and will warn if there appears to be an unmatched or missing FOR or NEXT. These are only warnings because APPLESOFT allows very unusual (and non-structured) use of the FOR and NEXT commands. For example, it is possible to repeat the same NEXT command several times as follows: "IF...THEN NEXT i,j". APLUS expects one and only one NEXT for each FOR that it encounters. The "MISSING NEXT" and "MISSING FOR" warnings can be ignored if you understand why they occur. APLUS does not make any changes to FOR...NEXT loops during "&CONVERSion".

The "SELECT", "(value)", and "(OTHERWISE)" Commands:

The "SELECT" command is one of the two "case"-type constructs supported by APLUS.

The "SELECT(variable)" statement must be given first. It tells APLUS to expect one or more "(value)"'s and optionally one "(OTHERWISE)" to catch any "variable" values that didn't satisfy one of the prior "(value)"'s.

A "(value)" is any expression (constant, variable, formula, etc) that can be compared for equality with the "variable" specified in the "SELECT(variable)" statement. The expression must be enclosed in parenthesis. Each "(value)" is followed by one or more lines of COMMANDS and terminated with a "FIN". The commands following each given "(value)" will only be executed if the "variable" was not equal to any of the prior "(value)"'s and is equal to the given "(value)". Once any given "(value)" matches the "variable", all following "(value)"'s and the optional "(OTHERWISE)" will be skipped.

An "(OTHERWISE)" can be specified after all other "(value)"'s in a "SELECT" group. The COMMANDS following the "(OTHERWISE)" will be executed if the "variable" was not equal to any of the prior "(value)"'s. There is no "(value)" for the "(OTHERWISE)" command. It serves as a "catch all" type of "(value)" and if used, must come at the end of a "SELECT" group. The "(OTHERWISE)" and corresponding "FIN" are optional within any given "SELECT" group.

Each "SELECT(variable)" command must be terminated with a "FIN".

```
Format              Example             &CONVERTed
SELECT(variable)    SELECT( N$ )        REM
: ( value#1 )       : ("NAME1")         IF NOT ((N$)=("NAME1")) GOTO N1
: : COMMANDS        : : PRINT"NAME1"    PRINT"NAME1"
: ::FIN             : ::FIN             GOTO M
: ( value#2 )       : ("NAME2")         N1 IF NOT ((N$)=("NAME2")) GOTO N2
: : COMMANDS        : : PRINT"NAME2"    PRINT"NAME2"
: ::FIN             : ::FIN             GOTO M
: (OTHERWISE)       : (OTHERWISE)       N2 REM
: : COMMANDS        : : PRINT"OTHER"    PRINT"OTHER"
: ::FIN             : ::FIN             GOTO M
::FIN               ::FIN               M REM
```

Notes: The colons preceding each line in the above example should not be entered into your program. They will automatically be generated when you "&LIST" your program.

The "CASE", "(case)", and "(OTHERWISE)" Commands:

The "CASE" command is one of the two "case"-type constructs supported by APLUS.

The "CASE" statement must be given first. It tells APLUS to expect one or more "(case)"'s and optionally one "(OTHERWISE)" to catch any conditions that didn't satisfy one of the prior "(case)"'s.

A "(case)" is specified as a LOGICAL EXPRESSION enclosed in parenthesis. Each "(case)" is followed by one or more lines of COMMANDS and terminated with a "FIN". The commands following each given "(case)" will only be executed if the LOGICAL EXPRESSIONS in all prior "(case)"'s were false and the LOGICAL EXPRESSION for the given "(case)" is true. Once any given "(case)" is executed, all following "(case)"'s and the optional "(OTHERWISE)" will be skipped.

An "(OTHERWISE)" can be specified after all other "(case)"'s in a "CASE" group. The COMMANDS following the "(OTHERWISE)" will be executed if all of the other "(case)" LOGICAL EXPRESSIONS were false. There is no LOGICAL EXPRESSION for the "(OTHERWISE)" command. It serves as a "catch all" type of "(case)" and if used, must come at the end of a "CASE" group. The "(OTHERWISE)" and corresponding "FIN" are optional within any given "CASE" group.

Each "CASE" command must be terminated with a "FIN".

```
Format              Example             &CONVERTed
CASE                CASE                REM
: ( L.E.#1 )        : ( A<1 )           IF NOT ( A<1 ) GOTO N1
: : COMMANDS        : : PRINT"A<1"      PRINT"A<1"
: ::FIN             : ::FIN             GOTO M
: ( L.E.#2 )        : ( 1<=A<2 )        N1 IF NOT ( 1<=A<2 ) GOTO N2
: : COMMANDS        : : PRINT"1<A<2"    PRINT"1<A<2"
: ::FIN             : ::FIN             GOTO M
: (OTHERWISE)       : (OTHERWISE)       N2 REM
: : COMMANDS        : : PRINT"2<A"      PRINT"2<A"
: ::FIN             : ::FIN             GOTO M
::FIN               ::FIN               M REM
```

Notes: The colons preceding each line in the above example should not be entered into your program. They will automatically be generated when you "&LIST" your program.

The "DO procedure" Command:

The "DO" command allows you to reference subroutines by name. Any name is allowed for the "procedure" name. The only restriction is that the "DO procedure" statement must be enclosed in double quotes, begin with a "DO", and not contain any double quotes in the procedure name.

A fatal error will occur if the procedure referenced by the "DO procedure" is not defined elsewhere with a "TO procedure".

The same procedure name can be used repeatedly in several "DO procedure" commands.

```
Format              Example             &CONVERTed
"DO procedure"      "DO RUN TEST"       GOSUB N
                       ***
"DO procedure"      "DO RUN TEST"       GOSUB N
                       ***
"TO procedure"      "TO RUN TEST"       N REM
COMMANDS            PRINT"TESTING"      PRINT"TESTING"
FIN                 FIN                 RETURN
```

The "TO define-procedure" Command:

Each named subroutine that you refer to with a "DO procedure" statement must be defined with a "TO procedure" command. The line number where the "TO procedure" occurs will be substituted into the "GOSUB" call when the corresponding "DO procedure" is "&CONVERTed".

There must be one and only one "TO procedure" for each procedure referenced in the rest of the program. A warning message will be generated if the procedure defined by the "TO procedure" command is not referenced anywhere else in the program.

A procedure is defined by a "TO procedure" statement followed by COMMANDS, followed by a "FIN". The "TO procedure" statement must be enclosed in double quotes, begin with "TO" and not contain any double quotes in the procedure name.

```
Format              Example             &CONVERTed
"DO procedure"      "DO RUN TEST"       GOSUB N
                       ***
"TO procedure"      "TO RUN TEST"       N REM
COMMANDS            PRINT"TESTING"      PRINT"TESTING"
FIN                 FIN                 RETURN
```

ERROR MESSAGES

PLEASE NOTE: A "FATAL ERROR" message early in your program can cause additional errors to occur later in your program. For example, a "MISSING FIN" error could offset the APLUS internal stack, causing a "STACK EMPTY" error later in the program.
PROBLEM CORRECTION WILL BE SIMPLIFIED IF <FATAL ERRORS> ARE CORRECTED ONE-AT-A-TIME, BEGINNING AT THE START OF YOUR PROGRAM.

PGM TOO BIG -- APLUS maintains an internal stack that begins at HIMEM and grows downward in memory towards your program as your logic becomes nested deeper and deeper. This message indicates that the APLUS stack pointer has collided with the top of your program. This message should only occur on systems with a marginal amount of RAM memory left after loading APLUS and the APPLESOFT program. The "PRINT FRE(Ø)" command can be used to determine how much memory is left. The problem is most likely to occur when "&CONVERTING".
SOLUTION: reduce either your program size or the depth of your deepest logic level.

STACK EMPTY -- APLUS maintains an internal stack (see PGM TOO BIG). This error should only occur after another "<FATAL ERROR>" has occured.
SOLUTION: correct the other error and this one should go away.

MISSING FIN -- A logical construct is not terminated properly earlier in the program. The cause of the problem may be many lines earlier than the occurance of the error message.
SOLUTION: carefully watch the indentation and to examine your logic to determine where the missing "FIN" should be placed.

MISSING ELSE -- A "WHEN" statement does not have the required matching "ELSE".
SOLUTION: insert "ELSE" in appropriate location.

MISSING FOR? -- This is only a WARNING message. The message indicates that you have at least one more "NEXT" command than you do "FOR" commands in your program.
SOLUTION: If this was what you were trying to do (it is valid in APPLESOFT, although very unstructured), then ignore this message. Otherwise, delete the extra "NEXT" or add the missing "FOR".

MISSING NEXT? -- This is only a WARNING message. The message indicates that you have at least one more "FOR" command than you do "NEXT" commands in your program.
SOLUTION: If this was what you were trying to do (it is valid in APPLESOFT, although very unstructured), then ignore this message. Otherwise, delete the extra "FOR" or add the missing "NEXT".

MISSING CASE/SELECT -- APLUS has found either a "(case)", "(value)", or "(OTHERWISE)" statement that was not preceded by either a "CASE" or a "SELECT(variable)" statement.
SOLUTION: add the missing "CASE" or "SELECT(variable)".

UNMATCHED FIN OR ELSE -- APLUS has found either a "FIN" or an "ELSE" that was not preceded by the proper APLUS command.
SOLUTION: review your logic and determine if the "FIN" or "ELSE" should be deleted. If you decide that the offending "FIN" or "ELSE" should stay, add the appropriate preceding commands.

UNDEFINED "DO -- APLUS cannot find a "TO procedure" that corresponds to the "DO procedure" that you are trying to use.
SOLUTION: first check the procedure list to see if any procedure names are close to the name that you are trying to use. Next, examine the names closely to determine whether there are any extra blanks or non-printing control characters in either the "DO or the "TO procedure names.

UNREFERENCED "TO -- APLUS cannot find a "DO procedure" statement that references this procedure. THIS IS ONLY A WARNING.
SOLUTION: if you desire to have an unreferenced procedure (for development, debugging, etc.), ignore this message. Otherwise, delete the procedure or add a "DO procedure" reference.

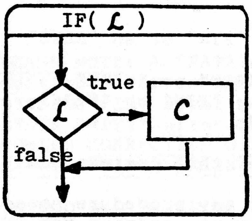
DUPLICATE "TO -- APLUS has found two or more "TO procedure" definitions for a given procedure name. The line where the duplicate definition occurs will be displayed.
SOLUTION: rename or delete one of the procedures.

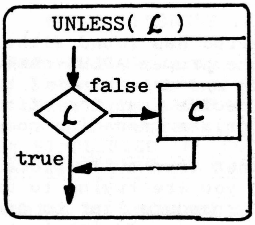
ADDENDUM

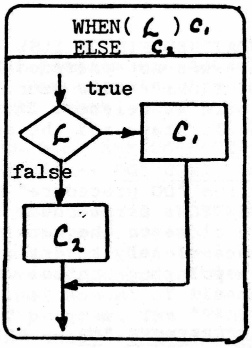
THE FOLLOWING POKES CAN BE USED TO ALLOW "&LIST"ING PROGRAMS TO APPLE SERIAL CARDS THAT ARE SET TO PRINT MORE THAN 40 COLUMNS PER LINE:

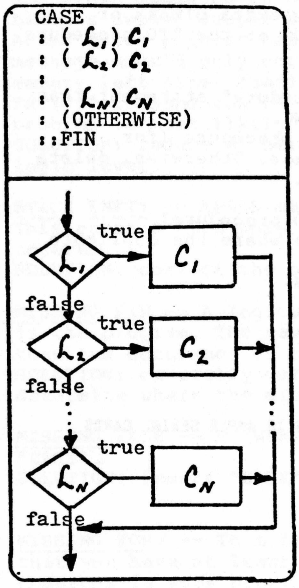
```
POKE 1784+SLOT#, LINE WIDTH
POKE 2040+SLOT#, 33
```

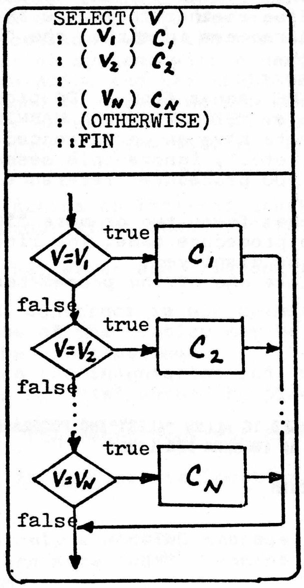
APLUS Structured-Basic Precompiler Summary









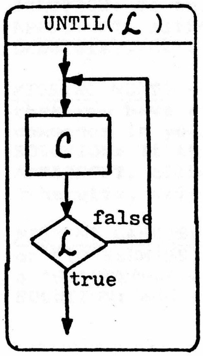


"DO carry-out-
 action"
"TO carry-out-
 action"

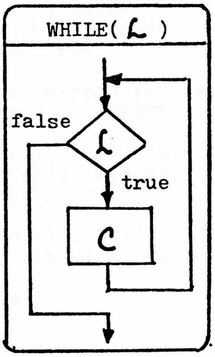
Note: Place an END before the 1st "TO" statement.

Note: (OTHERWISE) is optional in CASE & SELECT groups.

Legend:
$\mathcal{L}$ = Logical Expression
 A=B, A B, etc.
$\mathcal{C}$ = Applesoft commands
$\mathcal{V}$ = A variable or
 variable value



Note: OTHERWISE can be used as a "catchall" condition or variable value in SELECT and CASE groups.



11/28/79 SS

NOTES:

NOTES: